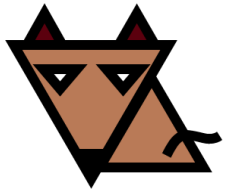
ネットワーク量子化の調査

Survey論文[Guo2018]が元ネタ

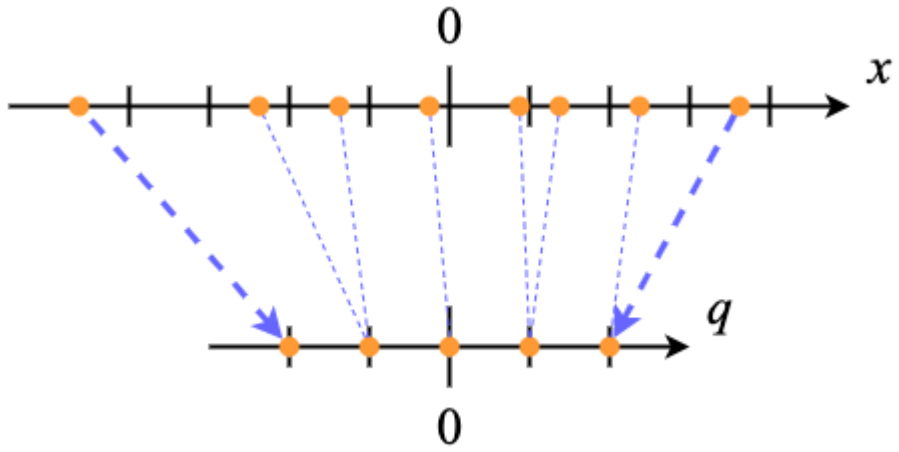
内容



- 量子化とは
- 量子化の目的
- 何を量子化するのか
- 量子化の分類
- 量子化ネットワークと学習
- 量子化ネットワークの精度

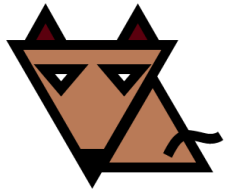
量子化とは

- 本来の意味：　連続量を離散量に置き換えること
- ここでは：　浮動小数点数を離散量に置き換えること



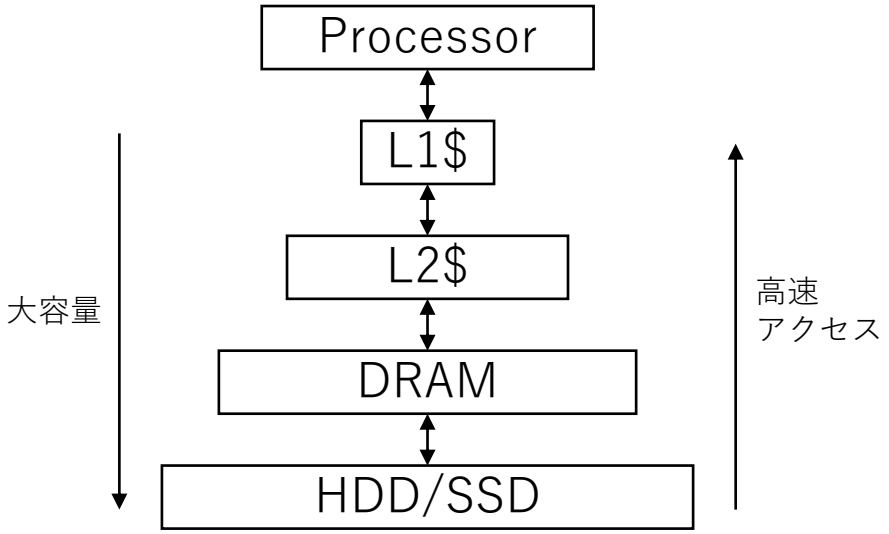
https://lab.mo-t.com/blog/quantization-frameworks

量子化の目的



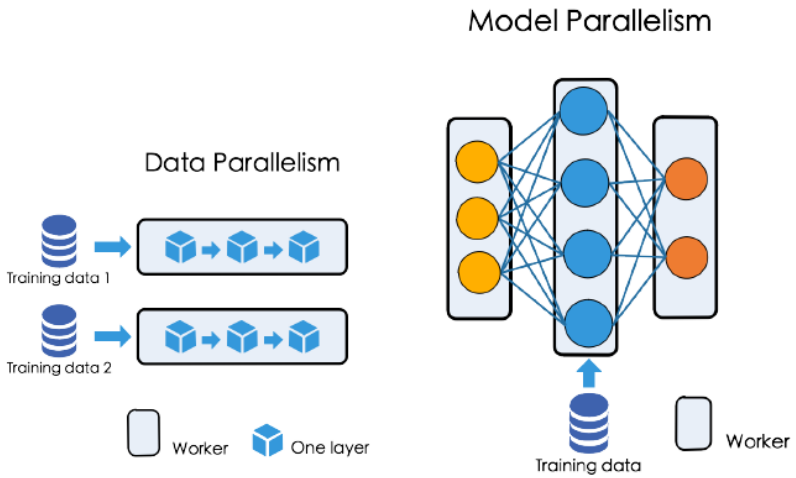
- ほとんどの論文の目的は**係数量・データ量を減らすこと**
- あまり速度向上を謳っていない（アーキ依存で比較しづらい、演算数が減るとも限らない）

メモリ階層



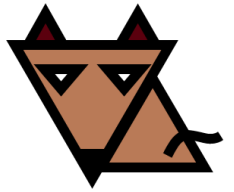
なるべく上の階層に多くの
係数／データを収めたい

分散学習

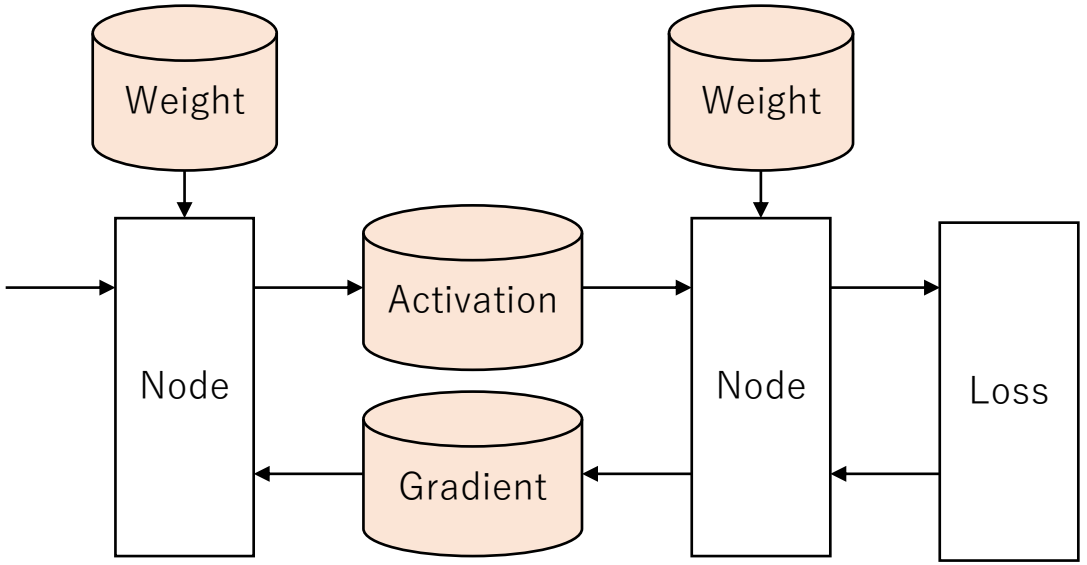


演算ノード間のデータ転送
を減らしたい

何を量子化するのか

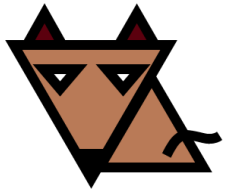


- Weight, Activation, Gradientの組合せ（or どれか）を量子化



Components	Benefits from Quantization	Challenges
Weights	Smaller model size Faster training and inference Less energy	Hard to converge with quantized weights Require approximate gradients Accuracy degradation
Activations	Smaller memory footprint during training Allows replacement of dot-products by bitwise operations Less energy	“Gradient mismatch” problem
Gradients	Communication and memory savings in parallel network training	Convergence requirement

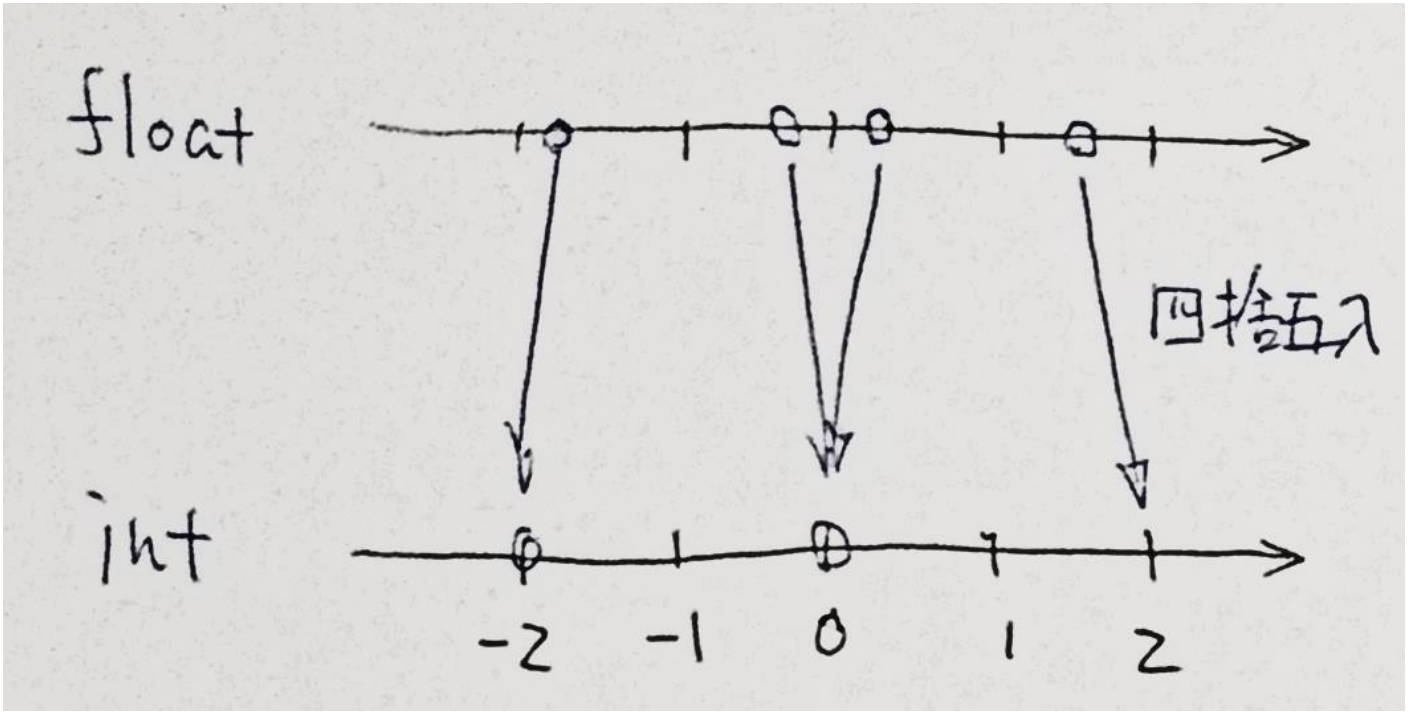
量子化の分類



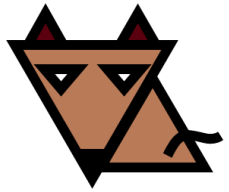
- 整数への置き換え
- 固定小数点数への置き換え
- 別の範囲・スケールへの置き換え
- とても狭い範囲への置き換え
- 不均一な離散化
- 最適化問題の解としての量子化
- 確率的な離散化

整数への置き換え

- 浮動小数点数を整数に置き換え
- 四捨五入・五捨六入・切り捨て・切り上げ等

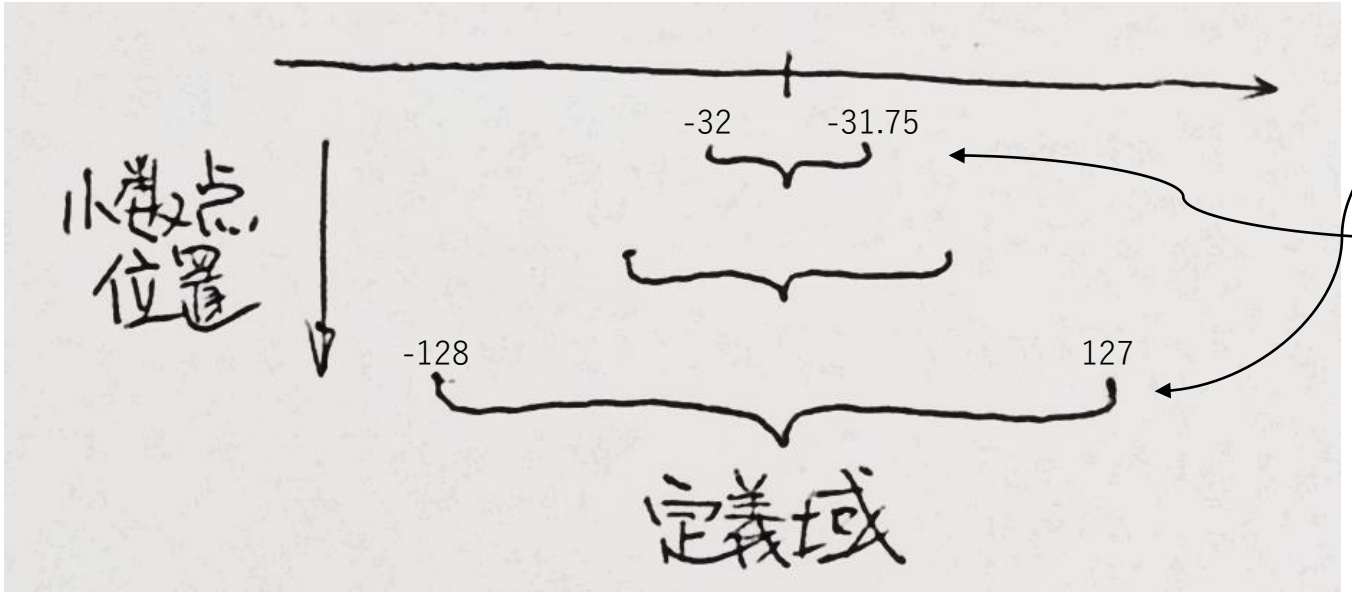


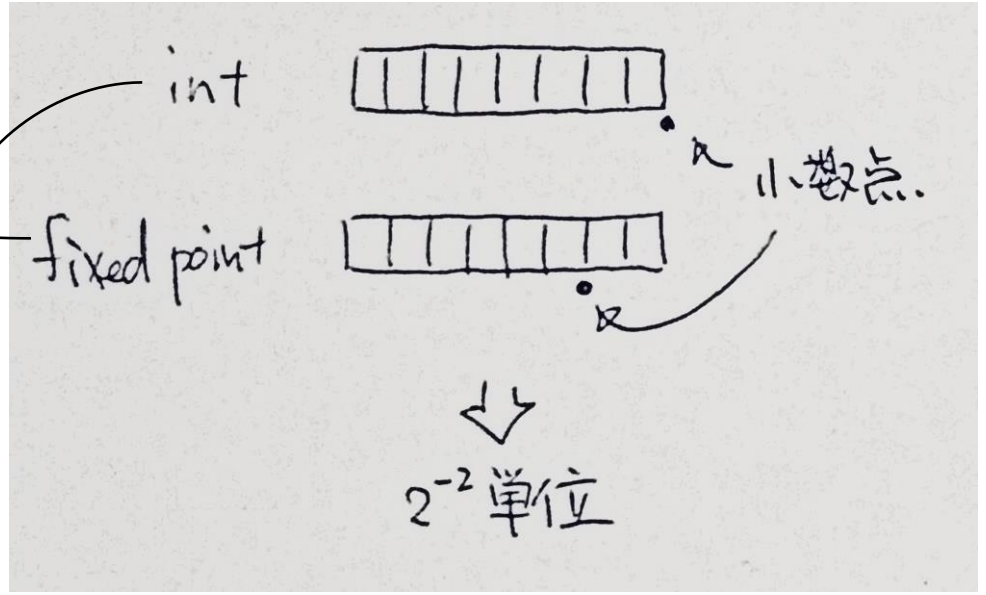
固定小数点数への置き換え



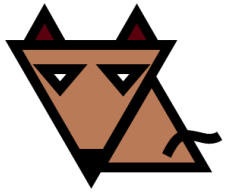
- 入力データの範囲（ダイナミックレンジ）に合わせて、小数点位置を決める

8bitの場合



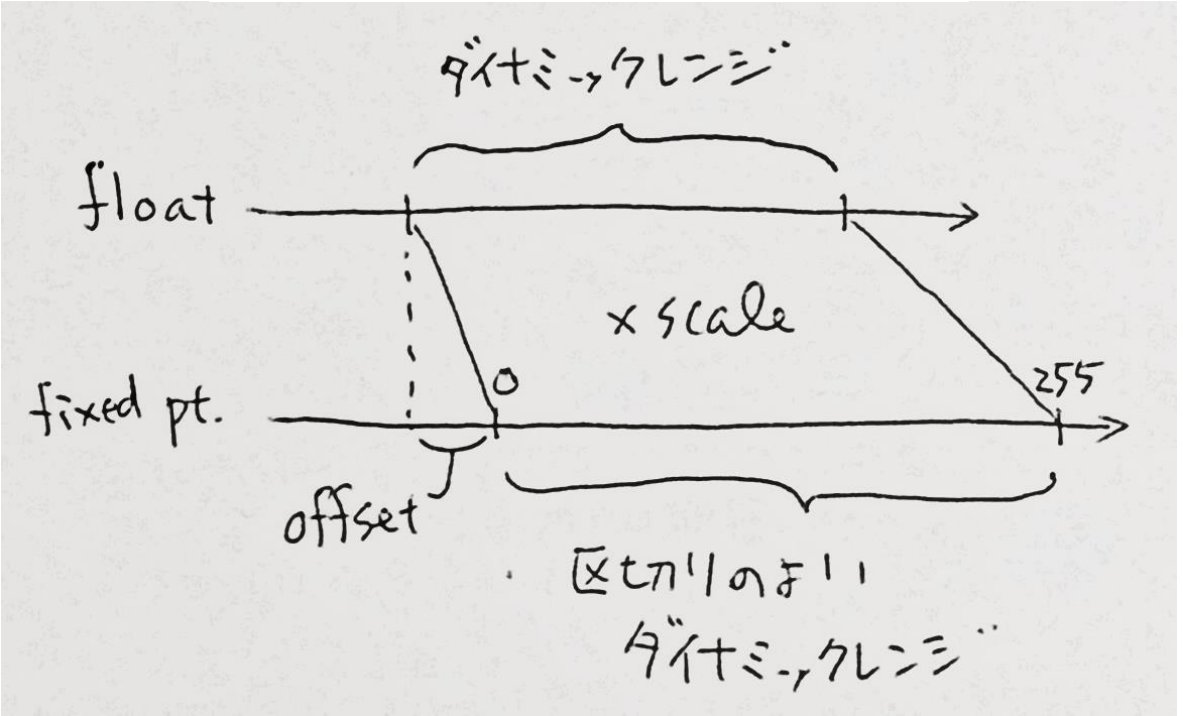


別の範囲・スケールへの置き換え

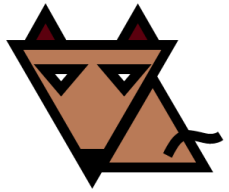


- 入力ダイナミックレンジを「区切りのよいダイナミックレンジ」（例えば0～255）に置き換えることで、使えるダイナミックレンジを最大限に活用する

$$q = \mathrm{offset} + \mathrm{round}\left(\frac{x}{\mathrm{scale}}\right)$$

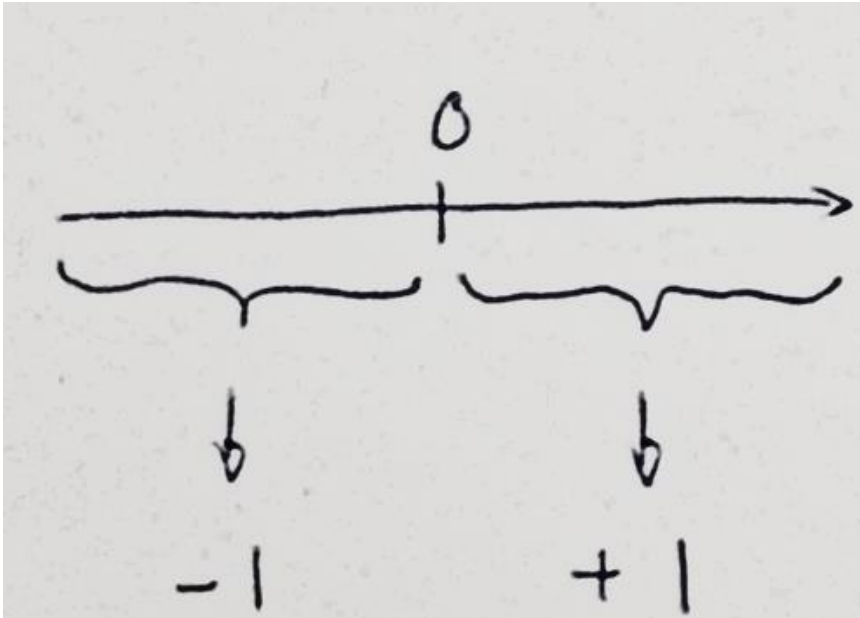


とても狭い範囲への置き換え



- 2値化(Binarization)、3値化(Ternarization)

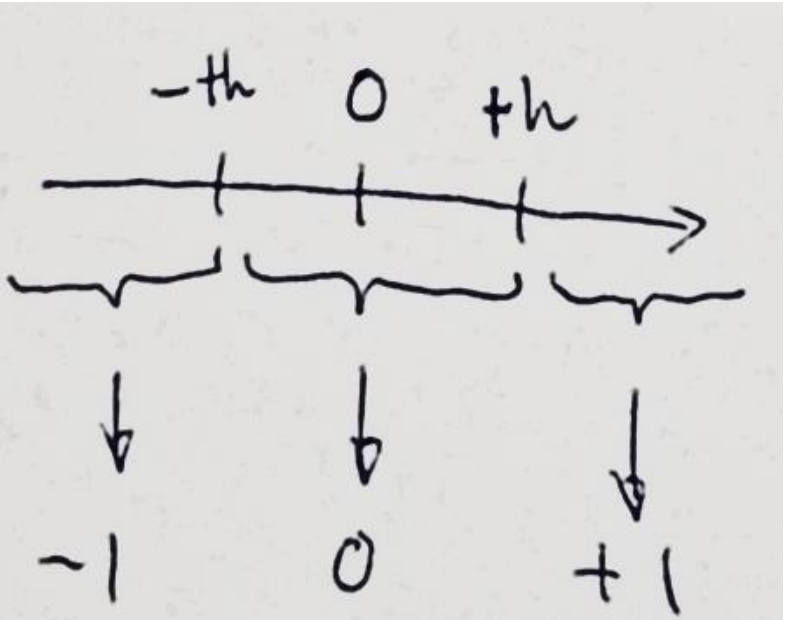
2値化(Binarization)



$$x^b = \mathrm{Sign}(x) = \begin{cases} +1 & x \geq 0, \\ -1 & \text{otherwise} \end{cases}$$

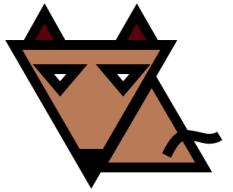
Binary Connect[Coubariaux2015]他

3値化(Ternarization)

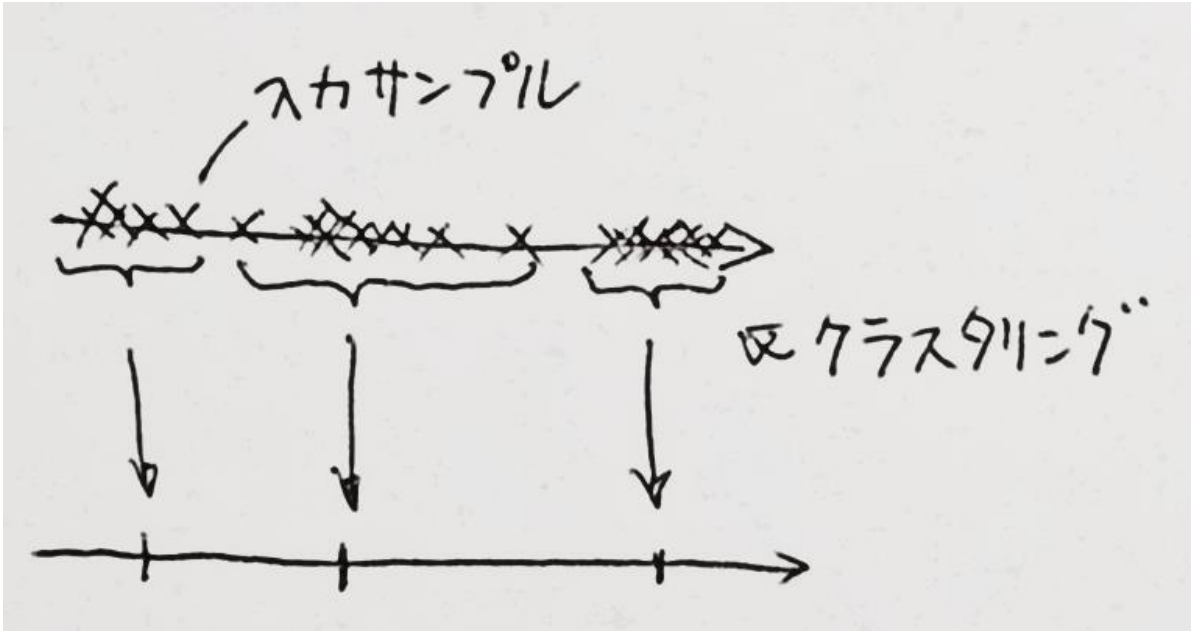


[Zhu2016]他

不均一な離散化

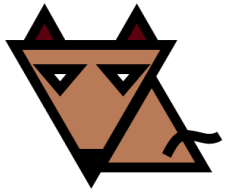


- 例えば、入力サンプルをクラスタリングし、クラスタ中心で離散化する



K-Meansで重みをクラスタリング[Gong2014]他

最適化問題の解としての量子化



- 最適化問題を解くことで離散値を求める

$$J(B, \alpha) = \|W - \alpha B\|^2 \tag{12}$$

where W is a real-value filter, B is the binary filter and α is a positive scaling factor. The optimal B and α are given as follows,

$$B^* = \text{Sign}(W), \quad \alpha^* = \frac{1}{n}\|W\|_{l_1} \tag{13}$$

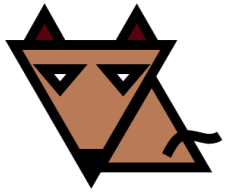
where n is the number of the elements in the filter. Interest-

XNOR-Net[Rastegari2016]

$$\alpha^*, W^{t^*} = \begin{cases} \text{argmin}_{\alpha, W^t} J(\alpha, W^t) = \|W - \alpha W^t\|_2^2 \\ \text{s.t.} \quad \alpha \geq 0, W_i^t \in -1, 0, 1, i = 1, 2, ..., n. \end{cases}$$

[Li2016]

確率的な量子化（１）

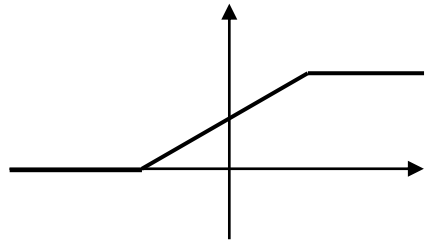


- ランダムに量子化

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x), \\ -1 & \text{with probability } 1 - p \end{cases}$$

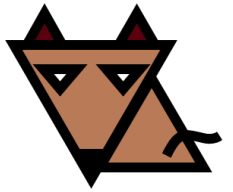
where σ is the ”hard sigmoid” function:

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2}))$$



Binary Connect[Coubariaux2015]

確率的な量子化（２）



- Bayesian Neural Networkの枠組みで量子化を考える

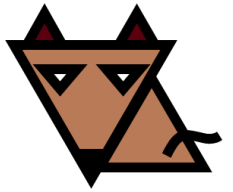
Assume a dataset $D = (x_n, y_n)_{n=1}^N$

$p(Y|X, \mathbf{w})$ be a parameterized neural network model

Bayesian neural networks, we want to estimate the posterior distribution of the weights given the data: $P(\mathbf{w}|D) = p(D|\mathbf{w})p(\mathbf{w})/p(D)$.

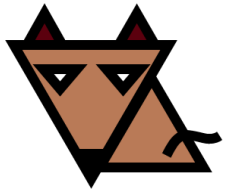
基本的なアイディアは、量子化の事前確率を用いて変分推論により事後確率を求める

量子化の分類まとめ



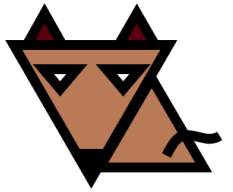
Types	Techniques	Descrition	Characteristics
Deterministic Quantization	Rounding	Using a quantization function to convert continuous values into discrete values	Simple to implement, can achieve good performance, often need to store the real values.
	Vector Quantization	Cluster the real values into subgroups	Simple to implement, can explore structural redundancy, only can be used to quantize pre-trained models.
	Quantization as Optimization	Convert the quantization problem into an optimization problem	Guaranteed to converge to a local minimum, more difficult to implement
Stochastic Quantization	Random rounding	Sampling quantized values according to given probabilities	Simple to implement, introduce noises as regularizers
	Probabilistic Quantization	Assume the weights are discretely distributed or learn multi-modal posterior disribution over weights	Consider structural redundancy, automatic regularization, easy to interpret

量子化ネットワークと学習



- 学習しながら量子化
- 学習してから量子化

学習しながら量子化



Algorithm 1 The training process of BinaryConnect

Notation: L is the number of layers in the network. $\mathbf{w}_{t-1}$ is the weight at time $t-1$. b_{t-1} is the bias at time $t-1$. a_k is the activation of layer k. E is the loss function.
Input: a mini-batch of data (inputs, labels), a learning rate η.
Forward pass:

- $\mathbf{w}_b \leftarrow \text{binarize}(\mathbf{w}_{t-1})$
- For $k = 1$ to L, compute a_k based on a_{k-1}, $\mathbf{w}_b$ and b_{t-1}.

Backward pass:

- Compute the gradient of the output layer $\frac{\partial E}{\partial a_L}$
- For $k = L$ to 2, compute $\frac{\partial E}{\partial a_{k-1}}$ based on $\frac{\partial E}{\partial a_k}$ and $\mathbf{w}_b$.

Parameter update:

- Compute $\frac{\partial E}{\partial \mathbf{w}_b}$ and $\frac{\partial E}{\partial b_{t-1}}$ based on $\frac{\partial E}{\partial a_k}$ and a_{k-1}
- $\mathbf{w}_t \leftarrow \text{clip}(\mathbf{w}_{t-1} - \eta \frac{\partial E}{\partial \mathbf{w}_b})$
- $b_t \leftarrow b_{t-1} - \eta \frac{\partial E}{\partial b_{t-1}}$

Binary Connect[Coubariaux2015]

STEも使われる

Forward: $x^b = \text{Sign}(x)$

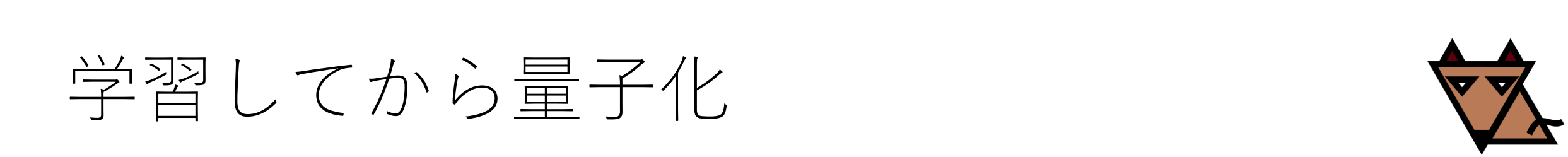
Backward: $\frac{\partial E}{\partial x} = \frac{\partial E}{\partial x^b} \mathrm{I}_{|x|\leq 1}$

where $\mathrm{I}_{|x|\leq 1}$ is an indicator function defined as,

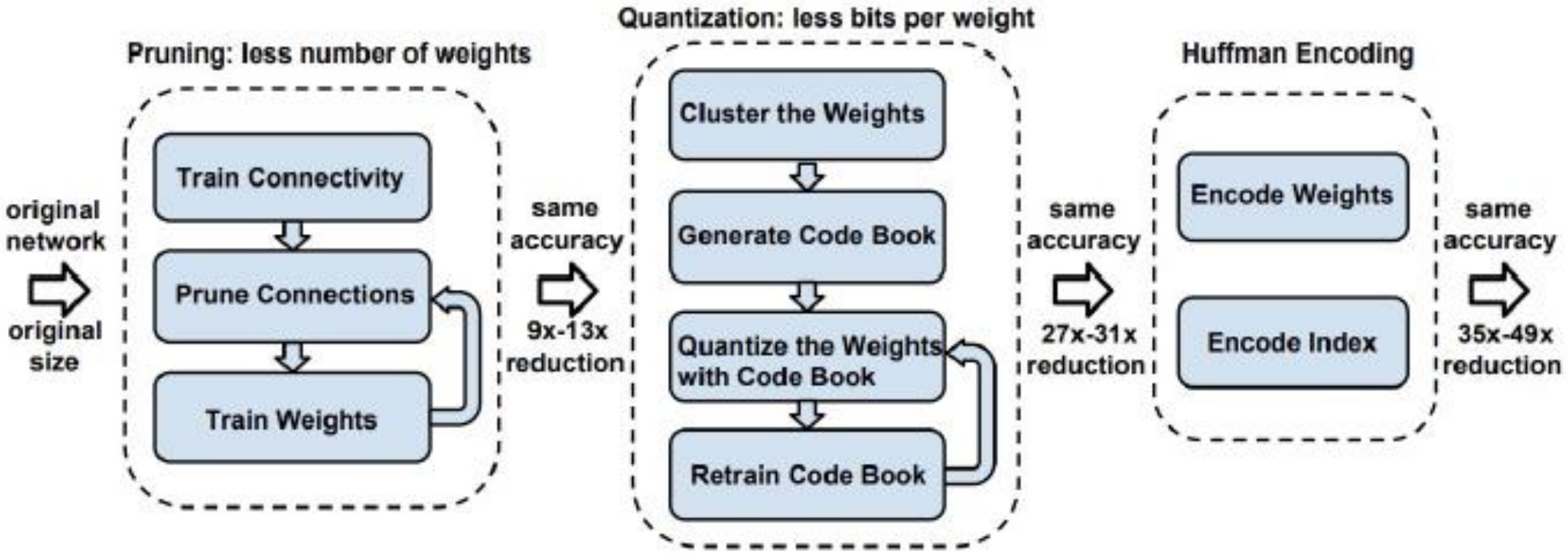
$$\mathrm{I}_{|x|\leq 1} = \begin{cases} 1 & |x| \leq 1, \\ 0 & \text{otherwise} \end{cases}$$

[Hinton2012]

学習してから量子化

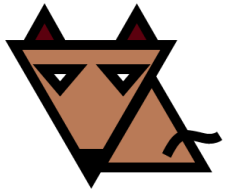


- 学習が楽で扱いやすい、Pre-trainedなネットワークでも使える



DeepCompression [Han 2015]

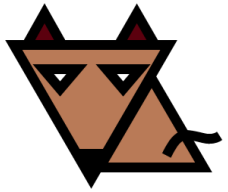
量子化ネットワークの精度



- 2値・3値でも浮動小数と変わらない性能が報告されている
- ただ、実際には性能が出ないデータもあるし、低bitの手法は学習が難しく扱いにくい
- 一般的には8bit程度への量子化ならあまり精度が落ちないようである（私見）

The recent quantized neural networks have achieved accuracy similar to their full-precision counterparts. For example, a binary network [Courbariaux *et al.*, 2015] can obtain 98.8% accuracy on the MNIST dataset. For large datasets such as ImageNet, a ternary network [Zhu *et al.*, 2016] can obtain comparable performance to the full-precision network.

参照



- [Coubariaux2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in Neural Information Processing Systems, pages 3123–3131, 2015.
- [Gong2014] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115, 2014.
- [Guo2018] Guo, Yunhui. A survey on methods and theories of quantized neural networks. arXiv preprint arXiv:1808.04752, 2018.
- [Han 2015] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.
- [Hinton2012] Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. Neural networks for machine learning. coursera, video lectures, 264, 2012b. 2012.
- [Li2016] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.
- [Rastegari2016] Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent neural networks with limited numerical precision. arXiv preprint arXiv:1608.06902, 2016.
- [Zhu2016] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. arXiv preprint arXiv:1612.01064, 2016.